ЛАБОРАТОРНАЯ РАБОТА 3

СОРТИРОВКИ ЧИСЛОВЫХ МАССИВОВ

Задание

Отсортировать по возрастанию числовой массив выбором, вставками и пузырьком. Получить зависимости временных затрат от размера массива. Временные затраты оценивать количеством сравнений и количеством обменов элементов массива.

СОРТИРОВКА ОБМЕНАМИ

Выполняется за n-1 проходов, где n – размер массива. В первом проходе первый элемент массива (с индексом 0) сравнивается с остальными элементами массива (с индексами 1…n-1) и, если предшествующий элемент больше последующего, они меняются местами. В последнем проходе сравниваются предпоследний и последний элементы массива. Следовательно, в каждом проходе количество сравнений фиксировано. Общее количество сравнений за сортировку равно n(n-1)/2.

Количество обменов зависит от состояния исходного массива. Если массив уже отсортирован, то обменов не будет. Это наилучший случай с точки зрения затрат машинного времени. Наихудший случай – при пересортировке массива, когда исходный массив отсортирован по убыванию. В этом случае каждое сравнение сопровождается обменом и, следовательно, количество обменов за сортировку равно n(n-1)/2.

СОРТИРОВКА ВЫБОРОМ

Эта сортировка требует также n-1 проходов. В первом проходе сравнениями первого элемента с остальными находят минимальный элемент и обменом помещают его на место первого элемента. В последнем проходе сравниваются предпоследний и последний элементы массива. Следовательно, в каждом проходе количество сравнений фиксировано. Общее количество сравнений за сортировку равно n(n-1)/2.

Число обменов всегда один на проход, следовательно, сортировка требует n-1 обменов. Наилучшего и наихудшего случаев не существует.

СОРТИРОВКА ВСТАВКАМИ

Выполняется за n-1 проходов. В первом проходе второй элемент массива (с индексом 1) сравнивается с первым элементом массива и, если второй оказывается меньше первого, первый сдвигается на место второго, а второй вставляется на место первого. В последнем проходе последний элемент массива сравнивается с предшествующими, начиная с предпоследнего, и вставляется между элементами, предыдущий из которых меньше, а последующий – больше последнего элемента. Среднее количество сравнений за сортировку - n(n-1)/4. Наилучший случай соответствует уже отсортированному массиву, когда количество сравнений равно n-1. Наихудший случай имеет место при пересортировке массива, при этом количество сравнений равно n(n-1)/2.

Обменов при сортировке вставками нет.

СОРТИРОВКА ПУЗЫРЬКОМ

В отличие от сортировки обменами, здесь сравниваются только соседние элементы и, если они стоят неправильно, меняются местами. В первом проходе сравниваются с соседним справа элементы с первого до предпоследнего. При этом сохраняется индекс первого из участвующих в последнем обмене элементов, что исключает избыточные просмотры массива. В следующем проходе просматривается часть массива с первого элемента по элемент с сохраненным индексом. Наилучший случай – массив уже отсортирован и нужен всего один проход с n-1 сравнениями. Наихудший случай – при пересортировке, когда потребуется n-1 проход с n(n-1)/2 сравнениями и n(n-1)/2 обменами.

РЕКУРСИВНЫЕ ФУНКЦИИ

В программировании рекурсия – вызов функции из этой же функции (простая рекурсия) или через другие функции (сложная рекурсия). Количество вложенных вызовов рекурсивной функции называют глубиной рекурсии. Рекурсивные функции реализуют рекурсивные алгоритмы.

Из вышеизложенного следует, что алгоритмы сортировок по существу являются рекурсивными алгоритмами, и поэтому сортировки естественно кодировать рекурсивными функциями. Однако большая глубина рекурсии приводит к переполнению стека, что является основным недостатком рекурсивных алгоритмов. Поэтому рекурсивные функции годятся для сортировок небольших по размеру массивов. Поскольку на практике обычно требуется сортировать данные больших объемов, то переходят к использованию итерационных алгоритмов, реализуемых циклами.

Проектирование приложения.

Часть данных для формирования массивов, а также функции сортировок инкапсулированы в классе, который вынесен в отдельный файл array.cs.

1. Запустите VS.
2. Создайте новый проект.
3. Перенесите на форму компонент menuStrip и создайте меню. Содержание меню отображено на рисунках 3.1 и 3.2. Создание меню не требует какого-либо программного кода и создается прямо на форме. Для создания события нажатия на пункт меню, щелкните по пункту два раза левой клавишей мыши.

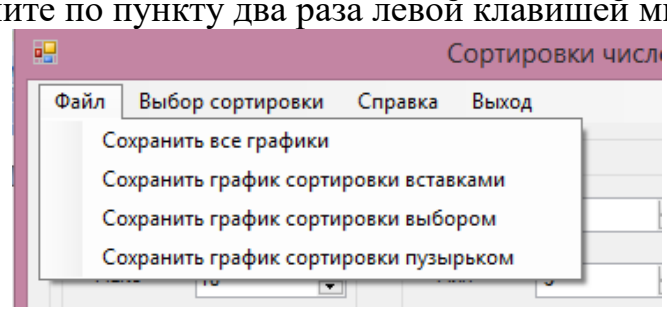

Рисунок 3.1 – Меню Файл

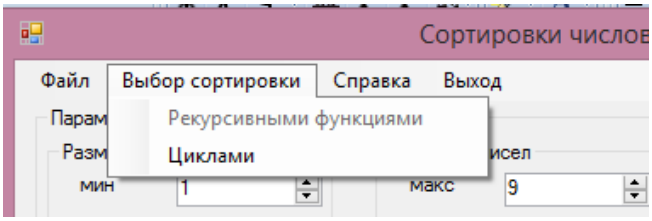

Рисунок 3.2 – Меню Выбор сортировки

4. Так же на форме присутствует компонент label, отвечающий за вывод сообщения при некорректном вводе диапазона чисел (см. лабораторную работу №1).

5. Визуальное разделение блоков Параметры массивов, Размеры и Диапазон чисел выполнено при помощи компонента groupBox.

6. Вкладки организованы с помощью компонента tabControl. Первая вкладка отображает исходный и отсортированные массивы. Для отображения использован компонент textBox (не забываем установить позицию true у свойства Multiline данного компонента). На второй вкладке расположены два компонента dataGridView (обязательна настройка автоподбора ширины и высоты ячейки по содержимому). На остальных трех размещены графики. Настройка визуального отображения графика частично реализована программно, частично, с помощью дерева настроек свойств компонента (коллекции свойств ChartAreas и Series).

7. По окончанию проектирования, форма примет вид, представленный на рисунке 3.3.

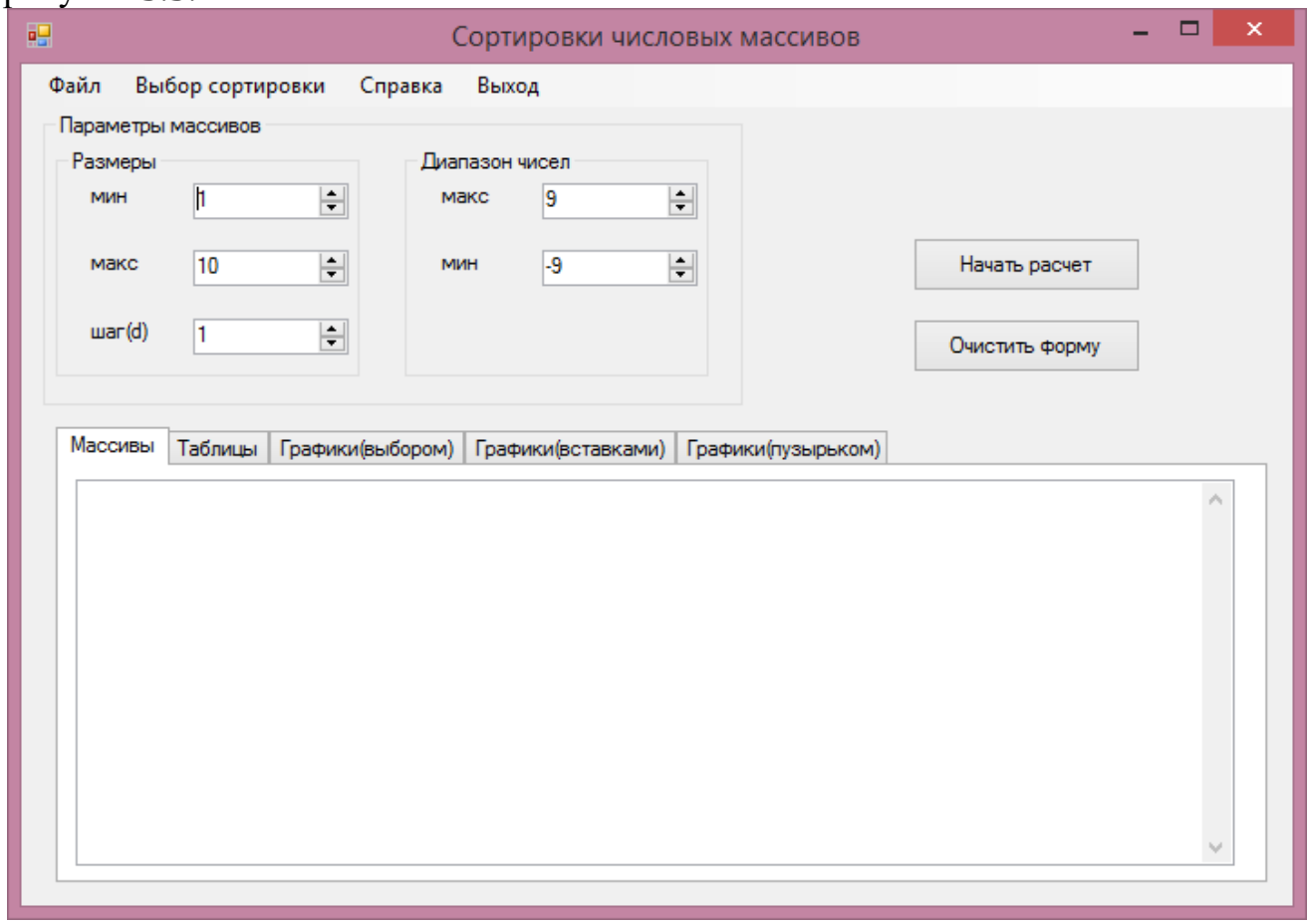

Рисунок 2.1 – Форма по окончанию проектирования

8. При нажатии на пункт меню Справка необходимо реализовать появление еще одного окна. Для этого выберите в меню Проект->Добавить новый элемент, а затем в появившемся окне выберите Форма Windows Form и дайте имя Form2 (рисунок 3.4). Поместите на форме компоненты label или textbox и введите информации о названии лабораторной работы, авторе и варианте работы. Пример реализации представлен на рисунке 3.5

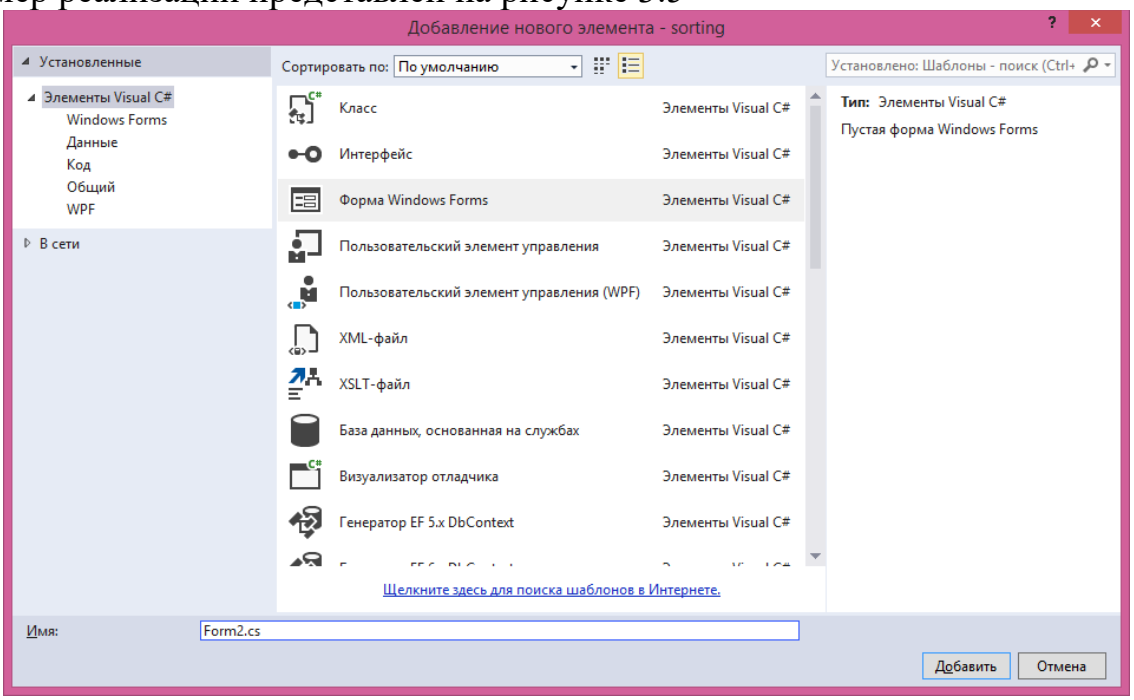

Рисунок 3.4 – Добавление новой формы

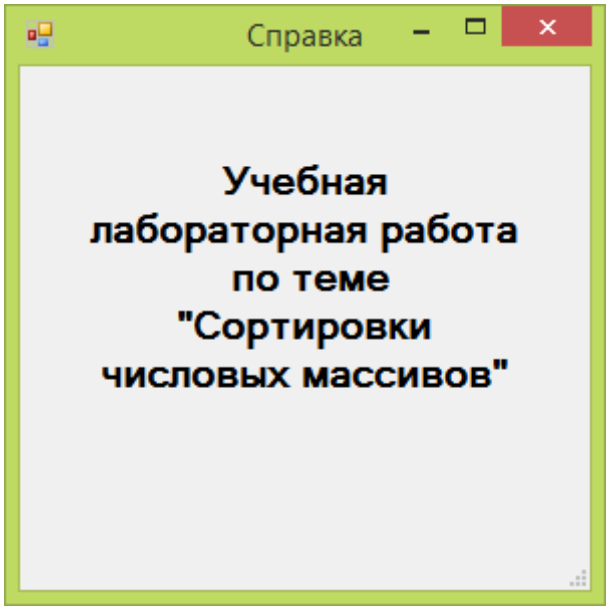

Рисунок 3.5 – Форма справки

9. Создайте файл с классом array.cs. Для этого выберите в меню Проект->Добавить класс, а затем в появившемся окне выберите Класс и дайте ему имя array.cs (рисунок 3.6). Новый файл появится в обозревателе решений вашего проекта.

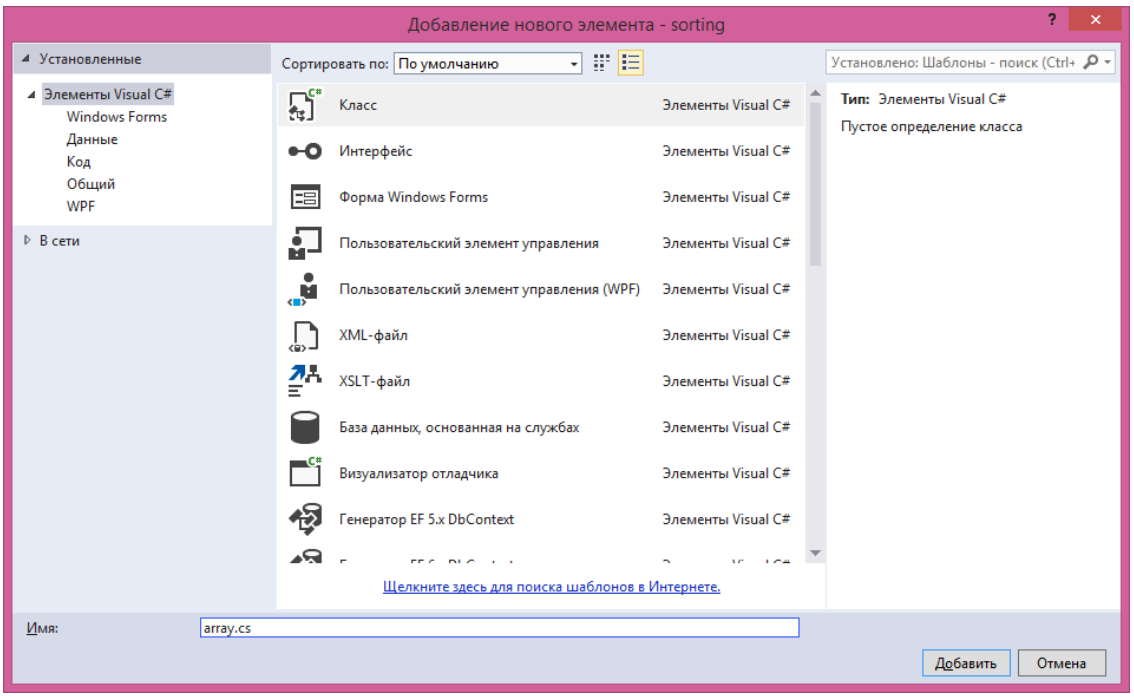

Рисунок 3.6 – Добавление файла класса

10. В файле array.cs введите следующий код класса:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace sorting
{
    class ArraySort
    {
        public ArraySort() //конструктор
        {
        }
       public int[] a;

       private static void swap(ref int x, ref int y)
        {
            int temp = x; x = y; y = temp;
        }

        public void SelectSort(int[] a, ref int sr, ref int obm)
        {
            int max;
            int length = a.Length;

            for (int i = 0; i < length - 1; i++)
            {
                max = i;

                for (int j = i + 1; j < length; j++)
                {
                    sr++;
                    if (a[j] > a[max])
                    {
                        max = j;
```

```
                    }
                }
                sr++;
                if (max != i)
                {
                    swap(ref a[i], ref a[max]);
                    obm++;

                }
            }

        }

        public void InsertSort(int[] a, ref int sr, ref int obm)
           {
              for (int i = 1; i < a.Length; i++)
       {
           int cur = a[i];
           int j = i;
           while (j > 0 && cur > a[j - 1])
           {
               sr++;
               a[j] = a[j - 1];
               j--;
           }
           a[j] = cur;
       }
              sr++;
           }

        public void BubbleSort(int[] a, ref int sr, ref int obm)
        {
            for (int i = 0; i < a.Length; i++)
            {
                for (int j = 0; j < a.Length - i - 1; j++)
                {
                    sr++;
                    if (a[j] < a[j + 1])
                    {

                        swap(ref a[j], ref a[j+1]);
                        obm++;
                    }
                }
            }

       }

    }
}
```

11. Создаем события обработки нажатия кнопок и пунктов меню. Код файла Form1.cs представлен ниже.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace sorting
{

    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            рекурсивнымиФункциямиToolStripMenuItem.Enabled = false;
            chart1.Series[0].BorderWidth = 3;
            chart1.Series[0].Color = Color.Red;
            chart1.Series[1].BorderWidth = 3;
            chart1.Series[1].Color = Color.Blue;
            chart1.Series[0].LegendText = "Сравнения";
            chart1.Series[1].LegendText = "Обмены";
            chart2.Series[0].BorderWidth = 3;
            chart2.Series[0].Color = Color.Red;
            chart2.Series[1].BorderWidth = 3;
            chart2.Series[1].Color = Color.Blue;
            chart2.Series[0].LegendText = "Сравнения";
            chart2.Series[1].LegendText = "Обмены";
            chart3.Series[0].BorderWidth = 3;
            chart3.Series[0].Color = Color.Red;
            chart3.Series[1].BorderWidth = 3;
            chart3.Series[1].Color = Color.Blue;
            chart3.Series[0].LegendText = "Сравнения";
            chart3.Series[1].LegendText = "Обмены";

        }
        public void output_textBox(int[] out_a, int n) //вывод массива в textBox
        {
            for (int i = 0; i < n; i++)
            { textBox1.Text += out_a[i] + " "; }
            textBox1.Text += Environment.NewLine;
        }

        public void output_dataGridView(int count, int sr, int obm, int vid_sort)//
вывод в таблицу кол-ва сравнений и обменов
        {
            dataGridView1.Rows[count].Cells[vid_sort].Value = sr;
            dataGridView2.Rows[count].Cells[vid_sort].Value = obm;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            button1.Enabled = false;
            dataGridView1.RowHeadersVisible = false;
            dataGridView1.ColumnCount = 4;
            dataGridView1.Columns[0].HeaderText = "Размер";
            dataGridView1.Columns[1].HeaderText = "Выбор";
            dataGridView1.Columns[2].HeaderText = "Вставки";
            dataGridView1.Columns[3].HeaderText = "Пузырек";
            dataGridView2.RowHeadersVisible = false;
            dataGridView2.ColumnCount = 4;
            dataGridView2.Columns[0].HeaderText = "Размер";
            dataGridView2.Columns[1].HeaderText = "Выбор";
            dataGridView2.Columns[2].HeaderText = "Вставки";
```

```
            dataGridView2.Columns[3].HeaderText = "Пузырек";
            dataGridView1.ColumnCount = 4;
            dataGridView2.ColumnCount = 4;
            if (numericUpDown4.Value < numericUpDown5.Value)
            { label8.Text = "Макс значение не м.б. меньше мин значения!"; return; }
            int count=0, n, sr=0,  obm=0;
            ArraySort sort_select = new ArraySort();
            ArraySort sort_insert = new ArraySort();
            ArraySort sort_bubble = new ArraySort();
            for (n = Convert.ToInt32(numericUpDown1.Value); n <=
Convert.ToInt32(numericUpDown2.Value); n += Convert.ToInt32(numericUpDown3.Value))
            {
                int[] base_a = new int[n];
                Random rand = new Random();
                for (int j = 0; j < n; j++)
                {
                    base_a[j] = rand.Next(Convert.ToInt32(numericUpDown5.Value),
Convert.ToInt32(numericUpDown4.Value));
                }


                textBox1.Text += "Исходный массив " + Environment.NewLine;
                output_textBox(base_a, n);

                dataGridView1.Rows.Add();
                dataGridView1.Rows[count].Cells[0].Value = n;
                dataGridView2.Rows.Add();
                dataGridView2.Rows[count].Cells[0].Value = n;



               sort_select.a = (int[])base_a.Clone();
               sr = 0; obm = 0;
               sort_select.SelectSort(sort_select.a, ref sr, ref obm);
               textBox1.Text += "Сортировка выбором " + Environment.NewLine;
               output_textBox(sort_select.a, n);
               output_dataGridView(count, sr, obm, 1);
               chart1.Series[0].Points.AddXY(n, sr);
               chart1.Series[1].Points.AddXY(n, obm);



               sort_insert.a = (int[])base_a.Clone();
               sr = 0; obm = 0;
               sort_insert.InsertSort(sort_insert.a, ref sr, ref obm);
               textBox1.Text += "Сортировка вставками " + Environment.NewLine;
               output_textBox(sort_insert.a, n);
               output_dataGridView(count, sr, obm, 2);
               chart2.Series[0].Points.AddXY(n, sr);
               chart2.Series[1].Points.AddXY(n, obm);


               sort_bubble.a = (int[])base_a.Clone();
               sr = 0; obm = 0;
               sort_bubble.BubbleSort(sort_bubble.a, ref sr, ref obm);
               textBox1.Text += "Сортировка пузырьком " + Environment.NewLine;
               output_textBox(sort_bubble.a, n);
               output_dataGridView(count, sr, obm, 3);
               chart3.Series[0].Points.AddXY(n, sr);
               chart3.Series[1].Points.AddXY(n, obm);
```

```
            count++;

        }

    }

    private void button2_Click(object sender, EventArgs e)
    {
        dataGridView1.Rows.Clear();
        dataGridView1.Columns.Clear();
        dataGridView2.Rows.Clear();
        dataGridView2.Columns.Clear();
        textBox1.Text = "";
        button1.Enabled = true;
        цикламиToolStripMenuItem.Enabled = true;
        chart1.Series[0].Points.Clear();
        chart1.Series[1].Points.Clear();
        chart2.Series[0].Points.Clear();
        chart2.Series[1].Points.Clear();
        chart3.Series[0].Points.Clear();
        chart3.Series[1].Points.Clear();

    }

    private void сохранитьВсеГрафикиToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        сохранитьГрафикСортировкиВставкамиToolStripMenuItem_Click(sender, e);
        сохранитьГрафикВортировкиВыборомToolStripMenuItem_Click(sender, e);
        сохранитьГрафикСортировкиПузырькомToolStripMenuItem_Click(sender, e);

    }

    private void сохранитьГрафикСортировкиВставкамиToolStripMenuItem_Click(object
sender, EventArgs e)
    {

     using (SaveFileDialog saveGr1 = new SaveFileDialog())
      {
          saveGr1.Title = "Сохранить график как ...";
          saveGr1.Filter = "*.jpg|*.jpg";
          saveGr1.AddExtension = true;
          saveGr1.FileName = "Insert";
          if (saveGr1.ShowDialog() == System.Windows.Forms.DialogResult.OK)
          {
              chart2.SaveImage(saveGr1.FileName,
System.Windows.Forms.DataVisualization.Charting.ChartImageFormat.Png);

          }
      }


    }

    private void сохранитьГрафикВортировкиВыборомToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        using (SaveFileDialog saveGr = new SaveFileDialog())
        {
            saveGr.Title = "Сохранить график как ...";
            saveGr.Filter = "*.jpg|*.jpg";
```

```
                saveGr.AddExtension = true;
                saveGr.FileName = "Select";
                if (saveGr.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    chart1.SaveImage(saveGr.FileName,
System.Windows.Forms.DataVisualization.Charting.ChartImageFormat.Png);

                }
            }

        }

        private void сохранитьГрафикСортировкиПузырькомToolStripMenuItem_Click(object
sender, EventArgs e)
        {
             using (SaveFileDialog saveGr2 = new SaveFileDialog())
                    {
                        saveGr2.Title = "Сохранить график как ...";
                        saveGr2.Filter = "*.jpg|*.jpg";
                        saveGr2.AddExtension = true;
                        saveGr2.FileName = "Bubble";
                        if (saveGr2.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
                        {
                            chart3.SaveImage(saveGr2.FileName,
System.Windows.Forms.DataVisualization.Charting.ChartImageFormat.Png);

                        }
                    }
        }

        private void цикламиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            button1_Click(sender, e);
            цикламиToolStripMenuItem.Enabled = false;

        }

        private void справкаToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form2 help = new Form2();
            help.ShowDialog();

        }

        private void выходToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }

    }
}
```

12. По окончанию проектирования и свертке методов код должен выглядеть следующим образом (рисунок 3.7).

```
public partial class Form1 : Form
{
    public Form1()...

    public void output_textBox(int[] out_a, int n) //вывод массива в textBox...

    public void output_dataGridView(int count, int sr, int obm, int vid_sort)// вывод в таблицу кол-ва сравнений и обменов...

    private void button1_Click(object sender, EventArgs e)...

    private void button2_Click(object sender, EventArgs e)...

    private void сохранитьВсеГрафикиToolStripMenuItem_Click(object sender, EventArgs e)...

    private void сохранитьГрафикСортировкиВставкамиToolStripMenuItem_Click(object sender, EventArgs e)...

    private void сохранитьГрафикСортировкиВыборомToolStripMenuItem_Click(object sender, EventArgs e)...

    private void сохранитьГрафикСортировкиПузырькомToolStripMenuItem_Click(object sender, EventArgs e)...

    private void цикламиToolStripMenuItem_Click(object sender, EventArgs e)...

    private void справкаToolStripMenuItem_Click(object sender, EventArgs e)...

    private void выходToolStripMenuItem_Click(object sender, EventArgs e)...

}
}
```

Рисунок 3.7 – Структура класса Form1

Тестирование и использование приложения

Пример выполнения приложения представлен на рисунках 3.8-3.10.

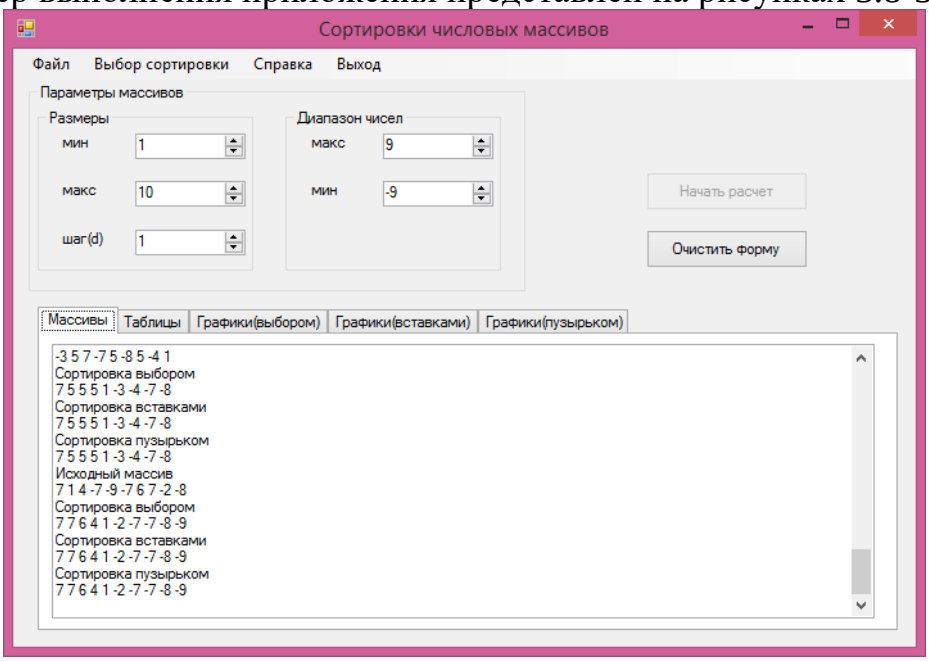

Рис.3.8 – Вкладка Массивы

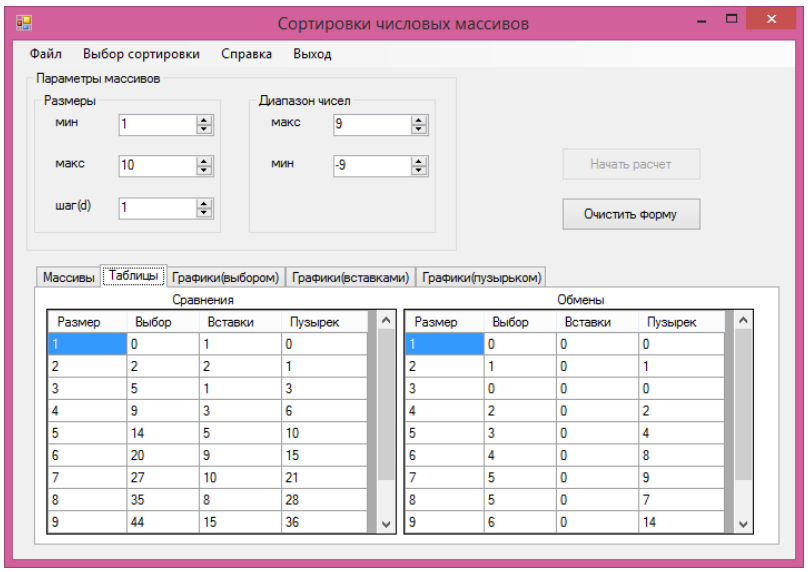

Рис.3.9 – Вкладка Таблицы

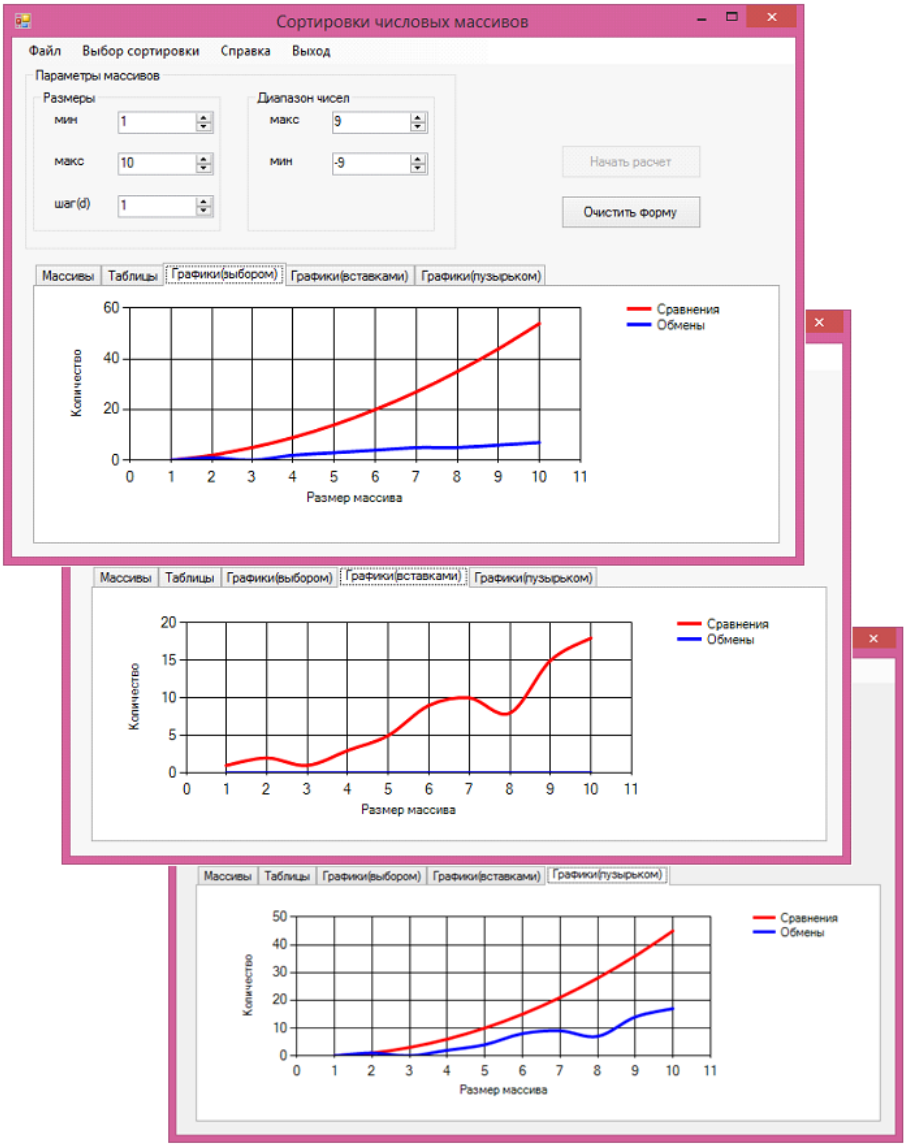

Рис.3.10 – Вкладки Графики

1. Запустите приложение на выполнение.
2. Выполните настройку параметров массива.
3. Нажмите на кнопку Начать Расчет.
4. Просмотрите результаты различных сортировок в форме текстовой информации, табличного представления и в виде графиков зависимостей количества сравнений и обменов в зависимости от размера массива. –
5. Нажмите на кнопку Очистить форму, а затем выберите пункт меню Выбор сортировки –>Циклами и убедитесь в правильности расчетов.
6. Сохраните графики по отдельности или все сразу, выбрав соответствующие пункты меню.
7. Нажмите на кнопки Справка и Выход, проверив их работоспособность.

Контрольные вопросы

1. Сравните алгоритмы сортировок рекурсивными функциями и циклами для способов: выбором, вставками, пузырьком.
2. Объясните алгоритм быстрой сортировки.
3. Объясните алгоритм сортировки Шелла.
4. Как по дереву объектов разместить на форме компоненты?
5. Объясните содержание класса *arraySort*.
6. Расскажите о правилах доступа вне и внутри класса к элементам в открытой и закрытой частях класса.
7. Какими возможностями располагает пользователь в файле реализации?
8. Где и как задаются параметры сортируемого массива? Как осуществляется связь между датчиками параметров и функциями сортировок?
9. Как оцениваются затраты машинного времени на сортировку массива?
10. Объясните ход зависимостей количества сравнений и обменов для сортировок.
11. Расскажите порядок работы с компонентом меню.
12. Как реализовано сохранение графиков в файл?
13. Как реализован вызов одной формы из другой?

Задания

1) Реализовать функциональную часть, представленную в методическом указании.
2) Реализовать одну из сортировок (вставка, выбор, пузырек с помощью рекурсивных функций), а также быструю сортировку или сортировку Шелла (по вариантам).
3) Оценить затраты машинного времени (алгоритм оценки создать самостоятельно) для пары сортировок (по вариантам). Если в паре, есть реализация, не созданная в пункте 1-2, то ее необходимо дописать

дополнительно. При оценке представить данные в табличной и графической форме. Реализовать возможность сохранения графиков в файл.

Задания на реализацию сортировок:

N вар.	Сортировки				
	Вставки	Выбор	Пузырек	Быстрая	Шелла
1	Рекурсия			*	
2		Рекурсия			*
3			Рекурсия	*	
4	Рекурсия				*
5		Рекурсия		*	
6			Рекурсия		*
7	Рекурсия			*	
8		Рекурсия			*
9			Рекурсия	*	
10	Рекурсия				*
11		Рекурсия		*	
12			Рекурсия		*

Задания на оценку затрат машинного времени:

ВсР – сортировка вставка (реализация с помощью рекурсии);
ВыбР – сортировка выбором (реализация с помощью рекурсии);
ПузР – сортировка пузырьком (реализация с помощью рекурсии);
ВсЦ – сортировка вставка (реализация с помощью циклов);
ВыбЦ – сортировка выбором (реализация с помощью циклов);
ПузЦ – сортировка пузырьком (реализация с помощью циклов);
Л – любая из представленных выше реализаций.

N вар.	Реализации для сравнения затрат машинного времени							
	ВсР↔ВыбР	ВсР↔ПузР	ПузР↔ВыбР	ВсЦ↔ВыбЦ	ВсЦ↔ПузЦ	ПузЦ↔ВыбЦ	Быстрая↔Л	Шелла↔Л
1	*						*	
2		*						*
3			*				*	
4				*				*
5					*		*	
6						*		*
N вар.	Реализации для сравнения затрат машинного времени							
	ВсР↔ВыбЦ	ВсР↔ПузЦ	ПузР↔ВыбЦ	ВсЦ↔ПузР	ВсЦ↔ВыбР	ПузЦ↔ВыбР	Быстрая↔Л	Шелла↔Л
7	*						*	
8		*						*
9			*				*	
10				*				*
11					*		*	
12						*		*